Mikko Kaipainen

Dokumenttien luokittelu luonnollisen kielen prosessointimenetelmillä

Tietotekniikan kandidaatintutkielma

7. heinäkuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Mikko Kaipainen

Yhteystiedot: mialkaip@student.jyu.fi

Ohjaaja: Tytti Saksa

Työn nimi: Dokumenttien luokittelu luonnollisen kielen prosessointimenetelmillä

Title in English: Document classification using natural language processing

Työ: Kandidaatintutkielma

Opintosuunta: Kaikki opintosuunnat

Sivumäärä: 14+0

Tiivistelmä: Tämä tutkimus käsittelee dokumenttien luokittelua luonnollisen kielen prosessoinnin menetelmillä. Tutkimuksessa esitellään vektoriavaruusmallit ja dokumentin luokittelu kolmella yleisellä ohjatun oppimisen menetelmällä. Tämän lisäksi tutkimuksessa selitetään luonnollisen kielen prosessoinnin kielitieteellistä taustaa.

Avainsanat: Luonnollisen kielen prosessointi, Vektoriavaruusmalli, Klusterointi, Luokittelija

Abstract: This study is about document classification methods using natural language processing. The study introduces vector space models and three common supervised learning models for document classification. The study also explains some linguistic background of natural language processing.

Keywords: Natural language processing, Vector space model, Clustering, Classifier

Sisältö

1 JOHDANTO 1

2 LUONNOLLISEN KIELEN PROSESSOINTI 2
2.1 Rationalistinen lähestymistapa 2
2.2 Empiristinen lähestymistapa 3

3 VEKTORIAVARUUSMALLIT 4
3.1 Dokumenttien klusterointi 4
3.2 Vektoriavaruusmallien painotusmetodit 5
3.3 Piilevä semanttinen indeksointi 5

4 DOKUMENTTIEN LUOKITTELU 7
4.1 Ominaisuuksien valinta 7
4.2 Naiivi Bayes-luokittelija 8
4.3 Tukivektorikone 8
4.4 K-lähimmän naapurin luokittelija 9

5 YHTEENVETO 10

LÄHTEET 11

1 Johdanto

Kun ihmisten välinen kommunikaatio ja tiedon levittäminen siirtyi suurelta osin verkkoon, syntyi valtava määrä luonnollisella kielellä kirjoitettua tekstiä. Siitä lähtien tämän datan hyödyntäminen on ollut keskeistä eri tietotekniikan sovellusaloilla. Viime vuosikymmeninä kehitettyjen tilastollisten menetelmien pohjalle rakennetut ratkaisut ovat käytössä monissa palveluissa ja sovelluksissa kuten hakukoneissa ja automaattisessa tekstin täydennyksessä.

Luonnollisen kielen prosessointi tutkimusalana pyrkii vastaamaan haasteisiin, jotka liittyvät luonnollisen kielen käyttöön tiedon lähteenä tietotekniikan sovelluksissa. Tietokoneohjelmat tavallisesti hyödyntävät rakenteellista tietoa, jossa data on sijoitettu tietokoneohjelman logiikan määräämälle paikalle. Rakenteellisia asiakirjoja käytettäessä ohjelman ei tarvitse tulkita mitä jokin symboli merkitsee tai mihinkä muihin symboleihin se liittyy juuri siinä kontekstissa, missä se sillä hetkellä esiintyy. Luonnollinen kieli sen sijaan on hyvin kontekstiriippuvaista. Asioita voi ilmaista monella tavalla ja sanojen ja lauseiden merkitykset ovat riippuvaisia niiden ympäröivästä tekstistä. Luonnollisen kielen prosessoinnin tarkoituksena on löytää säännöllisyyksiä sanojen esiintymistavoissa, joiden perusteella voidaan johtaa merkityksiä sanoille tai teksteille. Luonnollisen kielen prosessoinnin periaatteet ovat yhtä paikkaansa pitäviä jokaiselle luonnolliselle kielelle, mutta eri kielillä on toisistaan poikkeavia tapoja ilmaista lauseen jäsenten riippuvuuksia. Tässä tutkimuksessa käsitellään englannin kielisiä tekstejä.

Tutkimuksen toinen kappale käsittelee luonnollisen kielen prosessoinnin kielitieteellistä perustaa. Kappaleessa esitetään rationalistinen ja empiristinen lähestymistapa kielitieteeseen, ja kerrotaan miten ne ovat vaikuttaneet luonnollisen kielen prosessointiin. Kolmannessa kappaleessa kerrotaan vektoriavaruusmalleista ja niillä tehtävästä dokumenttien klusteroinnista. Neljännessä kappaleessa kerrotaan ohjatusta oppimisesta ja esitellään kolme yleistä luokittelijaa. Viidentenä kappaleena on yhteenveto tutkimuksesta.

2 Luonnollisen kielen prosessointi

Luonnollisen kielen prosessoinnissa pyritään ymmärtämään luonnollisen kielen ilmiöitä tietokoneiden avulla. Manning (1999) erottaa kielentutkimuksen karkeasti rationaaliseen ja empiristiseen lähestymistapaan.

2.1 Rationalistinen lähestymistapa

Kielitieteessä rationalistinen lähestymistapa perustuu oletukseen, että ihmisellä on jokin sisäänrakennettu tapa ymmärtää kieltä johon kaikki tavat ilmaista luonnollista kieltä perustuu. Rationalistista lähestymistapaa tukee muun muassa kielitieteilijä Noam Chomskyn argumentti ärsykkeen köyhyydestä, jonka mukaan lapsen kielellinen ärsykeympäristö ei yksin riitä selittämään, miten helposti lapsi oppii käyttämään äidinkieltään. Rationalistisessa lähestymistavassa kieltä pyritään mallintamaan kieliopilla, jolla erotetaan kieliopilliset lauseet.

Chomskyn tutkimuksessa (Chomsky 1956) arvioidaan millainen luonnollisen kielen kieliopin tulisi olla. Hänen mukaan kieliopin perusteella pitäisi pystyä selittämään lauseiden merkitykset yksiselitteisesti. Lauseille on kuitenkin monesti olemassa useampi kuin yksi kieliopillisesti oikea tulkinta. Chomsky esitti monitulkintaisuudelle ratkaisuksi transformaatiokieliopin, jossa jokainen kielellinen ilmaisu olisi johdettu joukosta ydinlauseita, joista muodostettaisiin kaikki kielen lauseet eri transformaatio-operaatioilla. Transformaatiokieliopin ongelmaksi hän osoitti sellaiset lauseet, joilla on useampi kuin yksi ydinlause. Esimerkiksi lause “Metsästäjien ammunta.” voidaan johtaa kahdesta ydinlauseesta: “Metsästäjät ampuvat” ja “Metsästäjät ammutaan.”. Monitulkintaisuusongelmasta huolimatta kielioppeja yritettiin käyttää nlp-sovelluksissa. Käytännössä kielioppien lisäksi käytettiin sovelluksen aihealueeseen liittyviä semanttisia sääntöjä. Esimerkiksi jonkun tietyn predikaatin kanssa saa esiintyä vain tiettyjä objekteja tai jollekin sanalle on valittu joku tietty merkitys monien joukosta. Tämä rajoitti sovellukset vain hyvin kapealle aihealueelle.

2.2 Empiristinen lähestymistapa

Empiristisessä lähestymistavassa hyödynnetään sanojen esiintymisfrekvensseistä johdettua tilastollista dataa. Tämä perustuu jakaumahypoteesiin, jonka mukaan sanoilla, jotka esiintyvät samoissa konteksteissa on samankaltaiset merkitykset. Harris perustelee jakaumahypoteesia tutkimuksessaan (Harris 1954) luonnollisen kielen säännöllisyydellä. Sanat eivät esiinny kielessä mielivaltaisesti, vaan niillä on jokin tietty esiintymistapa suhteessa muihin sanoihin. Siten Harrisin mielestä sanat olisivat ennustettavissa niiden ympäröivien sanojen perusteella. Harrisin ja muiden esiin nostama teoria kielen empiirisestä tutkimuksesta motivoi tilastollisten menetelmien kehittämiseen luonnollisen kielen prosessoinnissa.

3 Vektoriavaruusmallit

Vektoriavaruusmalleja käytetään sanojen, fraasien ja dokumenttien samanlaisuuden mittaamiseen. Vektoriavaruusmalleissa sanoista, fraaseista tai dokumenteista muodostetaan vektoreita, jotka sisältävät sanojen esiintymisfrekvensseihin liittyviä arvoja. Vektoreita vertaillaan erityyppisissä matriiseissa. Turney ja Pantel (2010) esittelivät kolme eri tyyppistä vektoriavaruusmallia: termi-dokumentti-matriisi, sana-konteksti-matriisi ja pari-esiintymistapa-matriisi. Dokumenttien luokittelun kannalta tärkein matriisityyppi on termi-dokumentti-matriisi, jossa kolumnivektoreina on dokumentit ja rivivektoreina sanaston sanat. Matriisin solujen arvot ovat sanojen esiintymisfrekvennsejä dokumenteissa. Sana-konteksti-matriisi ja pari-esiintymistapa-matriisi liittyvät sanojen ja fraasien semantiikkaan. Esimerkiksi sanojen synonyymejä voidaan löytää sana-konteksti matriisilla etsimällä sanoja, jotka esiintyvät samoissa konteksteissa.

3.1 Dokumenttien klusterointi

Tekstin luokittelu dokumenttivektoreilla esitettiin Gerard Saltonin “A vector space model for automatic indexing” tutkimuksessa (Salton, Wong ja Yang 1975). Dokumenttien indeksoinnissa dokumentille annetaan kuvaavia termejä jotka kertovat yleisellä tasolla mihin aiheisiin dokumentti liittyy. Saltonin ym. dokumenttivektoripohjaisessa automaattisessa indeksoinnissa dokumentin sanoista muodostettava sanajoukko indeksoi dokumentin. Tämä perustuu sana monijoukko hypoteesiin (bag of words hypothesis), jonka mukaan dokumenttien, joiden dokumenttivektorit ovat samankaltaiset, merkityssisällöt ovat samankaltaiset (from frequency to meaning). Tiedonhaun kannalta parasta on, että dokumenttivektorit klusteroituvat lähelle toisiaan ja klusterit ovat erillään toisistaan. Klusterit edustavat aiheeltaan samankaltaisia dokumentteja, joten tiivis klusteri kuvaa hyvin määriteltyä luokkaa. Mikäli dokumentit ovat levittäytyneet vektoriavaruuteen tasaisesti, luokkien välisten rajojen määrittäminen on hankalaa ja tiedonhaun tarkkuus laskee.

3.2 Vektoriavaruusmallien painotusmetodit

Yksinkertaisimmillaan indeksoinnissa käytettävät dokumenttivektorit sisältävät vain dokumentissa esiintyvien sanojen frekvenssit sellaisenaan, mutta usein käytetään jotain painottamis- ja tasoitusmetodia (smoothing method) paremman suorituskyvyn ja parempien tulosten saavuttamiseksi.

Painottamismetodeilla pyritään korostamaan niitä sanoja, joilla on erottava merkitys. Sanojen erottelukykyä voidaan mitata erotteluarvolla (Discrimination Value), jonka Salton ym. määritteli $DV_k = Q_k - Q$, jossa DV_k on erotteluarvo sanalle k, Q on dokumenttiavaruuden tiheys ja Q_k on dokumenttiavaruuden tiheys kun sana k on poistettu kaikista dokumenteista, joissa se esiintyy. Hyvälle erottelijalle dokumenttiavaruus on tiheämpi kun se on poistettu dokumenteista. Toisin sanoen, kun sana k esiintyy dokumenteissa, dokumenttivektorien väliset etäisyydet kasvavat, koska se erottelee dokumentit paremmin. Saltonin kokeiden perusteella sanoilla, jotka esiintyvät harvemmissa dokumenteissa, on parempi erottelukyky, mutta liian harvinaisilla sanoilla ei myös ole hyvää erottelukykyä.

Painotusmenetelmä tf-idf eli termifrekvenssi-käänteinen dokumenttifrekvenssi mukailee hyvin Saltonin kokeiden tuloksia. Menetelmässä painotetaan niitä dokumentissa usein esiintyviä sanoja, jotka esiintyvät harvoin koko dokumenttikokoelmassa. Muita painotusmetodeja ovat mm. vektorin pituuden normalisointi ja PMI. Tiedonhaku suosii pitkiä pitkiä dokumentteja, jota korjataan dokumenttivektorien pituuden normalisoinnilla. Pitkissä dokumenteissa todennäköisesti esiintyy enemmän merkittäviä indeksointitermejä, joka saa pitkät dokumentit vaikuttamaan merkittävämmiltä tiedonhaussa. Tosiasiassa dokumentin pituus ei kerro sen assosiaatiosta johonkin aiheeseen ja lyhyt dokumentti voi olla tiedonhakijalle yhtä kiinnostava kuin pitkä dokumentti.

3.3 Piilevä semanttinen indeksointi

Piilevä semanttinen indeksointi (latent semantic indexing) menetelmällä pyritään yhdistämään dokumentteja hakutermistöön perustuen dokumenttien piileviin (latent) merkityksiin. LSI-menetelmässä lasketaan termi-dokumentti matriisin pääakselihajotelma, jonka tuloksena on kolme eri matriisia.

Matriisille M SVD on $M = USV^*$. Saatujen matriisien perusteella voidaan määrittää kuinka samanlaisia termien esiintymistavat (patterns of occurance) kaikissa dokumenteissa ja miten samanlaisia dokumenttien termiprofiilit ovat. S matriisi on diagonaalimatriisi, joka vastaa piilevien merkitysten voimakkuutta. Matriisista S poistamalla pieniä arvoja ja niitä vastaavat rivit ja kolumnit matriiseista U ja V^* saadaan uusi matriisi M^*, joka on lähellä alkuperäistä matriisia M. $M*$ matriisin solujen arvot kuvaavat miten paljon termit liittyvät dokumentteihin. M^* on lähellä alkuperäistä M, mutta pienempi. Dokumenttien suhdetta hakutermistöön mitataan esittämällä hakutermit pseudodokumenttina, milloin se asettuu vektoriavaruuteen muiden dokumenttien tavoin.

Deerwester (Deerwester ym. 1990) pyrki LSI-menetelmällään parantamaan hakukoneindeksöinnin tarkkuutta. Indeksöinnillä tarkoitetaan dokumenttien yhdistämistä johonkin aiheeseen. Menetelmällä kyetään löytämään hakuun liittyviä dokumentteja vaikkei haussa käytettyjä termejä esiintyisikään dokumentissa. Vastaavasti menetelmä onnistuu erottamaan sellaiset dokumentit, joissa esiintyy hakutermejä, muttei liity haun piilevään merkitykseen. Deerwesterin mukaan LSI menetelmänä kykenee ratkaisemaan synonymiteettiin liittyvät ongelmat, mutta vain osittain onnistuu ratkaisemaan polysemisyyden ongelman. Polysemisellä termillä on useampi eri merkitys ja LSI antaa sanalle vain yhden pisteen vektoriavaruudessa perustuen sanan esiintymiseen kaikissa dokumenteissa.

4 Dokumenttien luokittelu

Dokumenttivektorien vertailu termi-dokumenttimatriisissa tuottaa vektoriavaruudessa havaittavia dokumenttiklustereita, joissa samankaltaiset dokumentit asettuvat lähelle toisiaan. Syntyneillä luokilla ei ole muuta tunnistavaa tekijää kuin muut dokumentit klusterissa. Joihinkin käyttötarkoituksiin, kuten tiedonlouhintaan, soveltuu paremmin luokittelija, jonka avulla erotetaan jokin tietty dokumenttiluokka muista. Menetelmät, joissa tiedetään mahdolliset luokat etukäteen perustuvat ohjattuun oppimiseen.

4.1 Ominaisuuksien valinta

Dokumentin luokittelumetodit kuten naivi Bayes ja tukivektorikone määrittävät dokumentille todennäköisyyden kuulua johonkin luokkaan siinä esiintyvien ominaisuuksien perusteella. Menetelmästä riippuen luokkia voi olla kaksi tai useampia. Ominaisuudet ovat sille luokalle tyypillisiä sanoja, joilla on joko binäärinen arvo tai reaalilukuarvo. Luokan ominaisuudet ovat valittu dokumenteista, jotka koetaan edustavan luokkaa riittävän hyvin. Ominaisuuksien valinnassa pyritään löytämään termit, joilla on suuri positiivinen tai negatiivinen assosiaatio luokkaan. Kokonaisia dokumenttivektoreita voidaan käyttää luokkien ominaisuuksien edustajina, mutta niiden suuri koko hidastaa luokittelijan suorituskykyä, joten usein on kannattavaa valita pienempi joukko ominaisuuksia.

Ominaisuuksien valintaan on laskennallisia menetelmiä. Esimerkiksi aikaisemmin esille nostettua tf-idf painotusmenetelmää voidaan käyttää ominaisuuksien valinnassa ottamalla ominaisuuksiksi ne termit, jotka saavat riittävän korkean arvon. Toinen käytetty ominaisuuksien valintametodi on bi-normal separation, joka luonnehditaan Baillargeon, Lamontagne ja Marceau (2019) tutkimuksessa sellaiseksi, jossa ominaisuuksille annetaan arvo sen perusteella, miten monta aitoa positiivista tulosta ne tuottavat binäärisessä luokittelutehtävässä verrattuna vääriin positiivisiin tuloksiin. Bi-normal separation antaa suurimman arvon sellaisille ominaisuudelle, joka esiintyy ainoastaan yhdessä luokassa, sillä se kykenee antamaan vain aitoja positiivisia tuloksia.

4.2 Naiivi Bayes-luokittelija

Eräs yksinkertaisimmista dokumentin luokittelijoista on naiivi Bayes luokittelija. Naiivi Bayes luokittelija laskee dokumentille todennäköisyyden kuulua luokkaan dokumentissa esiintyvien ominaisuuksien perusteella. Jokaiselle ominaisuudelle on annettu todennäköisyys kuulua johonkin luokkaan ja luokittelija laskee ominaisuuksien tulon kerrottuna luokan todennäköisyydellä jokaiselle luokalle. Dokumentti kuuluu luokkaan, jolla on suurin todennäköisyys. Esimerkiksi elokuva-arvosteluja luokiteltaisiin positiivisiin ja negatiivisiin laskemalla positiivisten ja negatiivisten arvostelujen suhde ja sanojen todennäköisyydet kuulua positiiviseen ja negatiiviseen luokkaan. Sanojen todennäköisyydet kuulua luokkaan laskettaisiin jakamalla sanan esiintymiskerrat luokassa kaikkien luokan sanojen ja koko sanaston summalla. Sanan esiintymiskertoihin voidaan lisätä 1 painottamaan tuntemattomia tapauksia. Jos jokin sana esiintyisi 2 kertaa positiivisissa arvosteluissa sen todennäköisyys positiiviselle luokalle olisi 2+1 jaettuna positiivisen luokan sanoilla plus arvostelujen sanoista koostuvan sanaston koko. Arvostelu luokiteltaisiin laskemalla todennäköisyys kummallekin luokalle harjoitusdatasta laskettujen arvojen perusteella ja luokka valittaisiin suurimman tuloksen perusteella.

Luokittelija perustuu oletukseen, että ominaisuudet esiintyvät tekstissä itsenäisesti vaikuttamatta muiden ominaisuuksien esiintymisiin. Oletus ominaisuuksien itsenäisyydestä on siinä mielessä naiivi, koska voimme olettaa, että tekstissä esiintyvät sanat vaikuttavat muihin tekstissä esiintyviin sanoihin. Tästä huolimatta naiivi Bayes suoriutuu luokittelutehtävistä lähes yhtä hyvin kuin uudemmat menetelmät kuten tukivektorikone. Fabrice Colas ja Pavel Brazdil havaitsi suorittamansa vertailun perusteella, että naiivi Bayes saavutti yleisesti hyvän suorituskyvyn verrattuna tukivektorikoneeseen sekä paremman suoritusnopeuden (Colas ja Brazdil 2006). Erityisesti luokittelijan koulutusvaiheessa naiivi Bayes saavutti huomattavasti paremman suoritusnopeuden kuin tukivektorikone, kun dokumenttien määrä kasvoi suureksi harjoitusdatassa.

4.3 Tukivektorikone

Tukivektorikoneessa hypertaso erottaa positiivisen ja negatiivisen vektorijoukon maksimoiden marginaalin, eli joukkojen etäisyyden, hypertasosta. Tukivektorikoneen opetuksessa ide-

aalinen tilanne on silloin kun molemmat joukot erottuvat selvästi toisistaan eikä yksikään dokumenttivektori asetu marginaaleille. Marginaaleille voidaan sallia tietty määrä vektoreita mikäli koetaan, että joukot ovat paremmin edustettuja. Tukivektorikone on lineaarinen luokitin, mutta sillä kyetään luokittelemaan myös dataa, joka ei erotu toisistaan lineaarisesti. Data kuvataan korkeampiulotteisessa avaruudessa milloin datalle voidaan löytää jokin hypertaso erottamaan ne toisistaan.

4.4 K-lähimmän naapurin luokittelija

K-lähimmän naapurin luokittelijassa (KNN) dokumentti luokitellaan k lähimmän naapurin perusteella. Esimerkiksi jos $k = 10$ niin algoritmi etsii kymmenen dokumenttia harjoitteludatasta, jotka ovat lähimpänä luokiteltavaa dokumenttia. Luokiteltava dokumentti kuuluu siihen luokkaan johonka kuuluvia dokumentteja algoritmi löysi eniten. Luokittelijassa voidaan käyttää myös painoja, jotka antavat suuremman painoarvon dokumenteille, jotka ovat lähimpänä luokiteltavaa dokumenttia.

K-lähimän naapurin luokittelijan tarkkuus on verrannollinen naivi bayesiin ja tukivektorikoneeseen. Luokittelijan tarkkuuteen vaikuttaa valittu k-arvo. Islam ym. (2007) tutkimuksessa vähiten vääriä luokituksia saatiin, kun *k*-arvo oli 5. Viittä pienemmillä arvoilla 3 ja 1 saavutettiin huomattavasti huonompi tarkkuus kuin myös suuremmilla arvoilla 11, 51 ja 101. Pienten *k*-arvojen väleillä luokitusten tarkkuus vaihteli paljon. Väärien luokitusten määrä pysyi tasaisena, kun k-arvo ylitti viidenkymmenen. Tutkimuksen mukaan parhaan *k*-arvon valinta on riippuu sovelluksesta, jossa luokittelijaa käytetään. Muissa tutkimuksissa kuten Brazdilin ja Colasin tutkimuksessa (Colas ja Brazdil 2006) paras tarkkuus saavutettiin huomattavasti suuremmalla *k*=45 arvolla.

Luokittelijan suoritusnopeuteen vaikuttaa eniten luokiteltavien tekstien määrä. Määrittääkseen luokiteltavan lähimmät naapurit, luokittelijan on laskettava etäisyys jokaisen dokumentin ja luokiteltavan välille. Näin ollen luokittelijan käyttämä aika kasvaa lineaarisesti dokumenttien määrän mukaan. Deng ym. (2016) tutkimuksessa esitettiin menetelmä, jossa luokiteltava datajoukko erotellaan pienemmiksi klustereiksi. Luokiteltavalle esineelle etsitään ensin lähin klusteri, jonka jälkeen k-lähintä naapuria etsitään klusterista.

5 Yhteenveto

Tässä tutkimuksessa käytiin läpi tekstin luokitteluun liittyviä periaatteita sekä muutama yleinen tekstin luokittelumenetelmä. Menetelmät jakaantuivat dokumenttien klusterointiin ja ohjatun oppimisen menetelmiin. Klusterointimenetelmänä kuvattiin G. A. Saltonin dokumenttien automaattinen indeksointimenetelmä sekä dokumenttivektoreihin liittyviä sievennysmenetelmiä. Dokumenttien luokittelu kappaleessa selitettiin, miten kolme yleistä luokittelijaa toimii. Tämän lisäksi kappaleessa kerrottiin ominaisuuksien valinnasta ja esiteltiin bi-normal separation menetelmä ominaisuuksien valintaan. Tutkimusta voisi laajentaa käsittelemään luokittelijoiden eri sovelluksia. Tässä tutkimuksessa esitellyt menetelmät eivät liity ainoastaan luonnollisen kielen prosessointiin, vaan ne ovat käytössä monilla eri sovellusaloilla. Laajennetussa tutkimuksessa voitasiin käsitellä esimerkiksi kohdennettua internetin kohdennettua mainontaa tai kohdennettua sisällön tarjontaa.

Lähteet

Baillargeon, Jean-Thomas, Luc Lamontagne ja Étienne Marceau. 2019. “Weighting Words Using Bi-Normal Separation for Text Classification Tasks with Multiple Classes”. Teoksessa *Canadian Conference on Artificial Intelligence,* 433–439. Springer.

Chomsky, Noam. 1956. “Three models for the description of language”. *IRE Transactions on information theory* 2 (3): 113–124.

Colas, Fabrice, ja Pavel Brazdil. 2006. “Comparison of SVM and some older classification algorithms in text classification tasks”. Teoksessa *IFIP International Conference on Artificial Intelligence in Theory and Practice,* 169–178. Springer.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer ja Richard Harshman. 1990. “Indexing by latent semantic analysis”. *Journal of the American society for information science* 41 (6): 391–407.

Deng, Zhenyun, Xiaoshu Zhu, Debo Cheng, Ming Zong ja Shichao Zhang. 2016. “Efficient kNN classification algorithm for big data”. *Neurocomputing* 195:143–148.

Harris, Zellig S. 1954. “Distributional structure”. *Word* 10 (2-3): 146–162.

Islam, Mohammed J, QM Jonathan Wu, Majid Ahmadi ja Maher A Sid-Ahmed. 2007. “Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers”. Teoksessa *2007 International Conference on Convergence Information Technology (ICCIT 2007),* 1541–1546. IEEE.

Manning, Schütze. 1999. *Foundations of statistical natural language processing.* MIT press.

Salton, Gerard, Anita Wong ja Chung-Shu Yang. 1975. “A vector space model for automatic indexing”. *Communications of the ACM* 18 (11): 613–620.

Turney, Peter D, ja Patrick Pantel. 2010. “From frequency to meaning: Vector space models of semantics”. *Journal of artificial intelligence research* 37:141–188.